4. Analízis modell kidolgozása

35 – negalt

Konzulens:
Ludmány Balázs

Csapattagok

Gutai Auguszta Matild	QWG59W	auguszta123@gmail.com
Fábián Csenge Zita	KGM10V	csengefabian@gmail.com
Ilosvay Viktória Brigitta	M27F3W	ilosviki@gmail.com
Szabó Marcell	LAUPQQ	szabomarcell.usa@gmail.com
Szabó Kinga Johanna	G003M6	szabo.kinga1999@gmail.com

2020-02-26

4. Analízis modell kidolgozása

4.1 Objektum katalógus

4.1.1 Ásó

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy használata során egy réteg hó helyett két rétegnyi havat tudjanak ellapátolni ugyanannyi munka ráfordítása mellett.

4.1.2 Búvárruha

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy ha később vízbe esnének, minden gond nélkül ki tudjanak menekülni maguktól.

4.1.3 Eszkimó

A játékosok egyik fajtája, speciális képessége az igloo építés. Az eszkimó testhője maximum 5 egységnyi lehet.

4.1.4 Étel

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy elfogyasztása során növelhessék testhőjüket.

4.1.5 Iglu

Az iglu egy jégmezőn elhelyezhető építmény. Megépítéséhez csak az eszkimó ért, de ezt követően minden mezőn tartózkodó játékos számára védelmet nyújt a vihar elől.

4.1.6 Játékos

A Játékosokat foglalja össze, akik lehetnek eszkimók vagy sarkkutatók. Ők a játék aktív résztvevői, a pályán mezőről-mezőre mozognak, körönként 4 egység munkát végezve próbálják elérni a csapat együttes célját, a jelzőrakéta összerelését és elsütését.

4.1.7 Jelzőfény

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

4.1.8 Jelzőrakéta

A patronból, pisztolyból és jelzőfényből összeszerelt tárgy, melyet a szereplőknek kulcsfontosságú feladatuk a játék megnyerésének célából összerakni és elsütni.

4.1.9 Jégmező

A mezőkből épül fel a játéktábla, a játékosok ezen mozoghatnak. A mezőkön lehet hó, és ennek szintje a viharok hatására növekedhet. A hó alatt a jégbe fagyva esetleg tárgyakat rejthet a mező. Egyes mezők teherbírása véges, ezekre nem léphet rá akárhány játékos, mert akkor felfordul.

4.1.10 Jégtábla (Játéktábla)

A mezők összességét szimbolizálja. A pálya mérete a játékosok számától függően különböző nagyságú lehet. A pálya négyzet alakú és tenger határolja, rajta szintén négyzet alakú mezők vannak. Ezek lehetnek jégmezők (stabil vagy instabil mezők) illetve lyukak.

4.1.11 Kötél

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy ha később egyik játékostársuk egy szomszédos mezőn vízbe esik, akkor ki tudják menekíteni. Mivel egy csapatként dolgoznak, fontos, hogy társaikat is megmentsék, különben ők sem fognak megmenekülni a zord sarki mezőről.

4.1.12 Lyuk

A jégtáblán lehetnek hóval fedett lyukak is. Ide lépve a játékos beleesik és megfullad, azaz vége a játéknak.

4.1.13 Patron

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

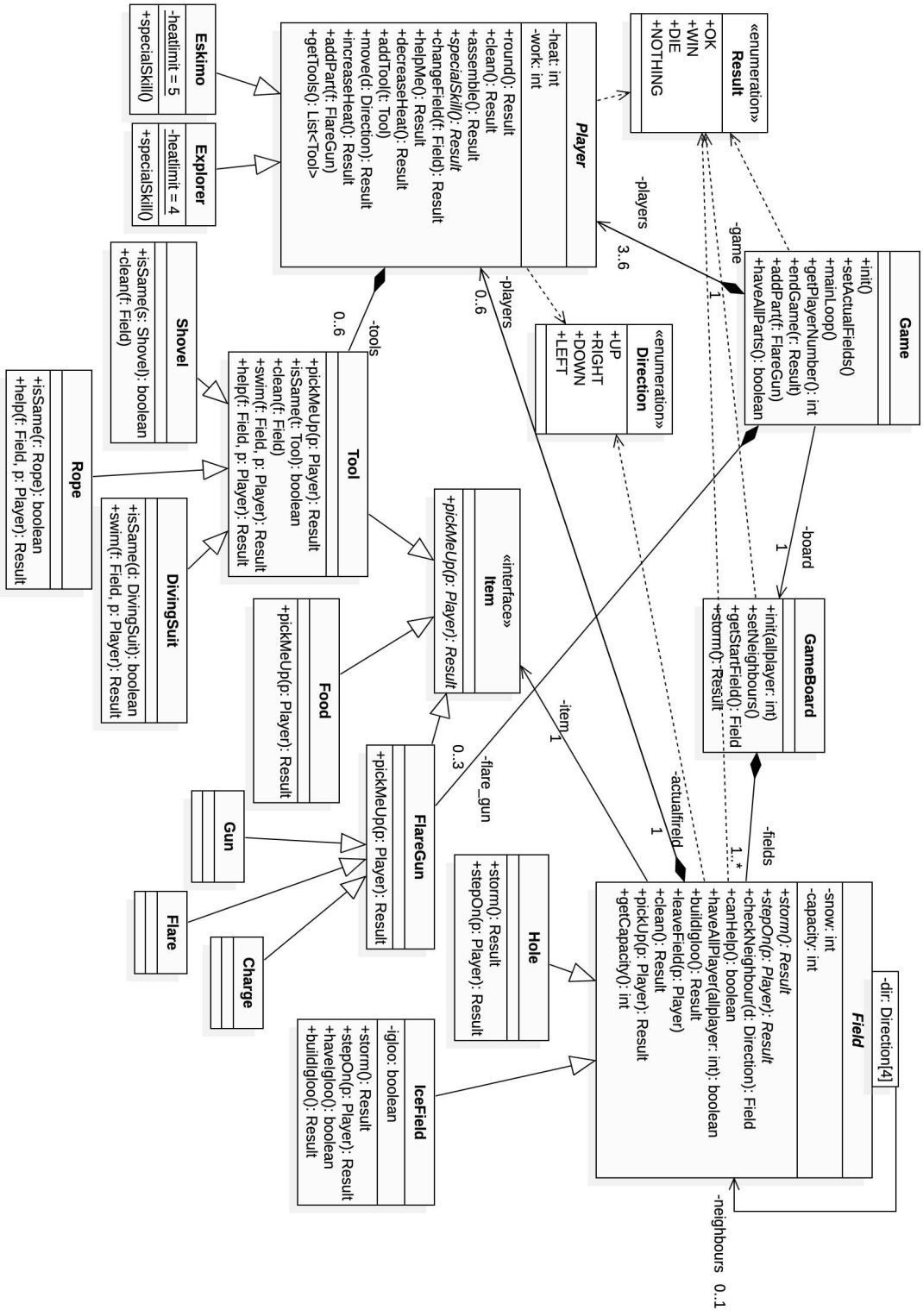
4.1.14 Pisztoly

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

4.1.15 Sarkkutató

A játékosok másik fajtája, akik speciális képessége a jégtáblák teherbírásának kiderítése. A sarkkutatók testhője maximum 4 egységnyi lehet.

4.2 Statikus struktúra diagramok



Az <<instantiate>> típusú denpencyket az osztálydiagram nem tartalmazza, mert azok nagyon átláthatatlanná tennék az ábrát.
Az osztálydiagramban szerepel három teljesen üres osztály: Flare, Gun és Charge. Ezeknek a projekt jelenlegi szakaszában nincs felelősségük, az osztálydiagramba csak a játék érthetősége miatt szúrtuk bele, tartalomuk és jelentősségük csak a grafikus rész kidolgozásánál lesz.

4.3 Osztályok leírása

4.3.1 Direction

Felelősség
Enumeration osztály, amely a négy irány közül vehet fel egy értéket.

Literálok

- **UP**: Fel
- **RIGHT**: Jobb
- **DOWN**: Le
- **LEFT**: Bal

4.3.2 DivingSuit

Felelősség
A búvárruha felvételének, illetve használatának kezelésére szolgáló osztály.

Ősosztályok

- Tool → DivingSuit

Interfészek

- Nincsenek.

Attribútumok

- Nincsenek.

Metódusok

- **boolean isSame(DivingSuit r)**: Mindig TRUE értékkel tér vissza (hiszen csak akkor hívódik meg, ha egy DivingSuit példány szeretné magát összehasonlítani vele).
- **Result swim(Field f, Player p)**: A Tool osztály swim(Field, Player) függvényének felüldefiniálása. A játékos által megadott irányt átadva meghívja az actualfield checkNeighbour(Direction) függvényét, ami visszatér az ott található mező referenciájával. Ha ez NULL érték lenne (tehát arra tenger van), akkor újra meg kell adni az irányt. Ha megkaptuk a választott szomszédos mező referenciáját, akkor ezt átadva meghívódik a paraméterben megkapott játékos changeField(Field) függvénye. Ennek a metódusnak a visszatérési értékével tér vissza a swim(Field, Player) függvény is.

4.3.3 Eskimo

Felelősség

Eszkimó karaktertípus esetén megadja a maximális testhő mértékét, illetve kezeli az eszkimó különleges képességét, tehát az igloo építésének menetét.

Ősosztályok

- Player → Eskimo

Interfészek

- Nincsenek.

Attribútumok

- **int heatlimit = 5**: Statikus attribútum. Az eszkimó testhő szintjének maximális számát adja meg.

Metódusok

- **void specialSkill()**: A Player osztályban lévő absztrakt függvény megvalósítása. Meghívja az actualfield attribútumban eltárolt Field-re a buildIgloo() függvényt, majd ennek visszatérési értékével (OK/ NOTHING) tér vissza ez a metódus is.

4.3.4 Explorer

Felelősség

Sarkkutató karaktertípus esetén megadja a maximális testhő mértékét, illetve kezeli a sarkkutató különleges képességét, tehát egy szomszédos mező teherbírásának vizsgálatát.

Ősosztályok

- Player → Explorer

Interfészek

- Nincsenek.

Attribútumok

- **int heatlimit = 4:** Statikus attribútum. A sarkkutató testhő szintjének maximális számát adja meg.

Metódusok

- **void specialSkill()**: A Player osztályban lévő absztrakt függvény megvalósítása. Először bekér egy irányt, majd erre meghívja a checkNeighbour(Direction) függvényt. Amennyiben ez NULL értékkel tér vissza, akkor a játékosnak újra meg kell adnia egy irányt. Amikor sikerül egy jó irányt megadni, tehát nem NULL visszatérési értéke lesz, akkor a visszakapott mezőre meghívja a getCapacity() függvényt.

4.3.5 *Field*

Felelősség

- Mezők kezelésére szolgáló absztrakt osztály. Segítségével a játékosok, illetve a vihar által a mezőn végzett tevékenységeket lehet megvalósítani.

Ősosztályok

- Nincsenek.

Interfészek

- Nincsenek.

Attribútumok

- **int snow:** Az adott mezőn lévő hóréteg mennyiségét tárolja.
- **int capacity:** Az adott mező teherbíró képességét tárolja el. Stabil jégtábla esetén a max játékosok száma, lyuk esetén pedig nulla az értéke.
- **Item item:** Tárolja, hogy a mezőn milyen tárgyat lehet felvenni.
- **Map<Direction, Field> neighbours:** Tárolja a 4 irányban elhelyezkedő mezőt.
- **Player players:** Tárolja, hogy az adott mezőn melyik játékosok vannak rajta.

Metódusok

- ***abstract Result storm()***: Absztrakt függvény. A Hole vagy az IceField osztály storm() függvénye kerül meghívásra.
- ***abstract Result stepOn(Player p)***: Absztrakt függvény. A Hole vagy az IceField osztály storm() függvénye kerül meghívásra.
- **Field checkNeighbour(Direction d)**: Az átadott irány alapján visszatér az abban az irányban levő objektum referenciájával. Ha arra tenger van, akkor ez az érték NULL lesz.
- **boolean canHelp()**: Akkor tér vissza OK értékkel, ha az adott mezőről megoldható a vízbe esett játékos vízből való kimentése. Ennek eldöntéséhez meghívja sorra összes rajta álló játékos getTools() függvényét, majd a megkapott Tool-okat tartalmazó listákra meghívja a help metódust. Ha legalább egy OK értékkel tér vissza, akkor ő is, különben pedig DIE-al.
- **boolean haveAllPlayer(int allplayer)**: Megvizsgálja, hogy az adott mezőn áll-e az összes játékos, tehát összehasonlítja a megkapott allplayer attribútumot a rajta állók számával. Ha ez megegyezik, akkor TRUE értékkel, ellenkező esetben FALSE értékkel tér vissza.
- **void buildIgloo(): Result**: Üres virtuális függvény, nem csinál semmit sem csak visszatér egy OK-al. Ez a függvény nem lesz meghívva a működés során, hanem ezen keresztül az IceField osztályban felülírt változata fog meghívódni.
- **void leaveField(Player p)**: Az attribútumban megkapott játékost kiveszi a players attribútumban tárolt játékosok referenciái közül (mivel a játékos elmozdult a mezőről).
- **Result clean()**: Megvizsgálja a snow attribútum értékét. Amennyiben ez nem nulla, akkor eggyel csökkenti az értékét, majd pedig OK-kal tér vissza. Ellenkező esetben nem történik meg a csökkentés, és a visszatérési érték NOTHING lesz.
- **Result pickUp(Player p)**: Megvizsgálja a rajta található hó mennyiségét. Ha ez nulla, és található rajta jégbe fagyott tárgy, akkor meghívja az Item osztály pickMeUp(Player) függvényét. Sikeres tárgyfelvétel esetén OK-kal tér vissza, egyébként pedig NOTHING-gal.

- **int getCapacity()**: Visszaadja az mező teherbíró képességét. (Az elkészült játékban valószínűleg ez a függvény nem fog visszatérni a teherbírás értékével, hanem csak megjeleníti a képernyőn azt. Most a grafikus elemek hiányában illetve a jobb érthetőség kedvéért használjuk ezt a függvényt így.)

4.3.6 FlareGun

Felelősség

A jelzőpisztoly alkotóelemeinek felvételével foglalkozó osztály.

Ősosztályok

- Nincsenek.

Interfészek

- Item

Attribútumok

- Nincsenek.

Metódusok

- **Result pickMeUp(Player p)**: Attribútumként átadva önmaga referenciáját meghívja a Player osztály addPart(FlareGun) függvényét.

4.3.7 Food

Felelősség

Étel felvételének kezelésére szolgáló osztály.

Ősosztályok

- Nincsenek.

Interfészek

- Item

Attribútumok

- Nincsenek.

Metódusok

- **Result pickMeUp(Player p)**: Meghívja a Player osztály increaseHeat() függvényét. Amivel az általa hívott metódus tér vissza, az lesz ennek a függvénynek is a visszatérési értéke.

4.3.8 Game

Felelősség

A játék elkezdésére/inicializálására, befejezésére, illetve a körök kezelésére szolgáló osztály. A viharok feltámadásnak valószínűségeit, és annak lebonyolításának kezdetét is ez az osztály kezeli.

Ősosztályok

- Nincsenek.

Interfészek

- Nincsenek.

Attribútumok

- **Player players**: Az adott játékosokat tárolására szolgáló tömb, melynek nagysága 3 és 6 között helyezkedik el (beleértve a határokat is).
- **GameBoard board**: A játékhoz tartozó játéktábla tárolására szolgál.
- **FlareGun flare_gun**: A jelzőrakéta alkotóelemeinek (GUN, FLARE, CHARGE) tárolására szolgál.

Metódusok

- **void init()**: A játék kezdetekor bekéri a játékosok számát majd sorra azoknak a karaktertípusát. Létrehozza a GameBoard-ot, majd meghívja az osztály init(Player) metódusát átadva neki a játékosok számát. Ezt követően a bekért adatok alapján létrehozza az eszkimókat illetve a sarkkutatókat reprezentáló osztályokat. Végül meghívja a setActualFields() metódust.
- **void setActualFields()**: A korábban létrehozott Player példányoknak állítja be az actualfield attribútumát. Ehhez lekéri a kezdő mező referenciáját a GameBoard-tól a getStartField() függvény segítségével.
- **void mainLoop()**: Ciklusban hívogatja meg a játékosok köreinek lezajlásáért felelős függvényeket. A ciklus minden lefutása során először egy adott valószínűség alapján eldönti hogy jön-e vihar. Ha jön, akkor meghívja a Gameboard storm() függvényét (ellenkező esetben ez a függvényhívás kimarad). Ha ez nem DIE-al tért vissza, akkor meghívja a Player osztály round() metódusát. Amennyiben ez bármikor DIE vagy WIN visszatérési értéket ad, vagy már korábban a storm() DIE-al tért vissza, akkor kilép a ciklusból. (Ciklusban maradáshoz OK visszatérési érték kell. NOTHING-nak itt nincs szerepe.). Végezetül az endGame(Result) függvény kerül meghívásra .
- **int getPlayerNumber()**: A players attribútumban tárolt játékosok számával tér vissza.
- **void endGame(Result r)**: A megkapott paraméter alapján eldönti, hogy hogyan végződik a játék, majd befejezi azt.
- **void addPart(FlareGun f)**: A flare_gun attribútum értékét változtatja meg, mégpedig úgy, hogy hozzáad egy új FlareGun példányt a listához.
- **boolean haveAllParts()**: Megvizsgálja, hogy a játékosok összegyűjtötték-e a három szükséges tárgyat (GUN, FLARE, CHARGE), tehát három elemet tartalmaz-e a flare_gun lista. Ha igen, akkor TRUE, máskülönben FALSE értékkel fog visszatérni.

4.3.9 GameBoard

Felelősség
Létrehozza, inicializálja, összefogja s egyben tárolja az összes mezőt. Ha az adott körben úgy adta a gép, hogy lesz hóvihar, akkor az előzőek mellett kezeli azt is, hogy melyik mezőkre fog leesni egy réteg hó.

Ősosztályok
- Nincsenek.

Interfészek
- Nincsenek.

Attribútumok
- **Field fields**: A táblához tartozó mezők tárolására szolgál.

Metódusok
- **void init(int allplayer)**: Az átvett játékos-szám alapján létrehozza a mezőket majd eltárolja azokat fields tömbben. Végül meghívja a setNeighbours() függvényt.
- **void setNeighbours():** A már elkészített Field-eknek beállítja a szomszédait a fields tömb alapján.
- **Field getStartField()**: Visszatér a bal felső sarokban lévő Field referenciájával. (Erről a mezőről fognak elindulni a játékosok).
- **Result storm()**: Eldönti egy adott valószínűség alapján minden egyes mezőre, hogy ott jön-e vihar. Ha jön, akkor meghívja annak a mezőnek(Field) a storm() függvényét. Futás végén, ha legalább egy mező storm() függvénye DIE-al tért vissza, akkor ő is DIE-al fog, különben pedig OK-kal.

4.3.10 Hole

Felelősség
Lyukak kezelésére szolgáló osztály. A vele kapcsolatos kimentési kísérletet is ez az osztály kezdi meg.

Ősosztályok
- Field → Hole

Interfészek
- Nincsenek.

Attribútumok
- Nincsenek.

Metódusok
- **Result storm()**: A *Field* osztályban lévő absztrakt függvény megvalósítása. Az adott mező snow attribútumának értékét megnöveli eggyel. Mindig OK-kal tér vissza.
- **Result stepOn(Player p)**: A *Field* osztályban lévő absztrakt függvény megvalósítása. Az attribútumként kapott Player példány helpMe() metódusát hívja meg, majd ennek a visszatérési értékével tér vissza a stepOn(Player) függvény is.

4.3.11 IceField

Felelősség

Jégtáblák/Jégmezők kezelésére szolgáló osztály. Egyrészt a jégmezők teherbírásának vizsgálatát végzi el, másrészt pedig vihar esetén kezeli, hogy ha az adott mezőn esik a hó, akkor az milyen kritériumok mellett (van-e igloo vagy nincs) milyen következményekkel jár (mezőn levő hórétegek számát mindig növeljük, viszont a testhő csökkentése csak az iglooval védetlen mezőkön történik meg).

Ősosztályok

- Field → IceField

Interfészek

Nincsenek.

Attribútumok

- **boolean igloo**: Megadja, hogy adott mező tartalmaz-e igloot. (Ha tartalmaz, akkor TRUE az értéke)

Metódusok

- **Result storm()**: A *Field* osztályban lévő absztrakt függvény megvalósítása. Az adott mező snow attribútumának értékét megnöveli eggyel. Amennyiben az adott mező nem tartalmaz igloo-t, akkor az ilyen mezőn álló játékosokra meghívja a decreaseHeat() metódust. Amivel ez a függvény visszatér, azzal tér vissza a storm() is.
- **Result stepOn(Player p)**: A *Field* osztályban lévő absztrakt függvény megvalósítása. OK értékkel tér vissza, ha az adott mezőn lévő játékosok számát még elbírja a jégtábla. Ellenkező esetben pedig DIE értékkel fog visszatérni.
- **boolean haveIgloo()**: Ha van igloo a mezőn, akkor TRUE-val tér vissza, ellenkező esetben pedig FALSE-szal.
- **Result buildIgloo()**: A *Field* osztály virtuális buildIgloo() metódusának a felüldefiniálása. Akkor hívódik meg, ha egy eszkimó igloot szeretne építeni a jégtáblán. Ha még nem volt igloo a mezőn, akkor az igloo attribútum értékét TRUE-ra állítja, majd OK értékkel tér vissza. Amennyiben volt igloo az adott, mezőn, akkor NOTHING lesz a visszatérési értéke.

4.3.12 Item

Felelősség

Interfész, amely a tárgyak egységes kezelését biztosítja, ez szolgál a tárgyak felvételének kezelésére.

Metódusok

- ***abstract Result pickMeUp(Player p)***: Absztrakt függvény. A FlareGun, a Food vagy a Tool osztály pickMeUp(Player) függvénye kerül meghívásra.

4.3.13 Player

Felelősség

Játékosok kezelésére szolgáló osztály. A játékosok munkájának és testhőjének vizsgálata mellett a körökben elvégezhető cselekvésekkel foglalkozik. Minden játékos köre addig tart, amíg a work attribútumának értéke nem csökken le nullára. Minden cselekvés, ami az adott esetben engedélyezett, az egy egység munkavégzéssel jár (például tárgy felvétele olyan mezőn, amin még van hóréteg nem engedélyezett, és ilyenkor ez nem is jár munkavégzéssel).

Ősosztályok

- Nincsenek.

Interfészek

- Nincsenek.

Attribútumok

- **int heat**: Az adott játékos testhő szintjének mennyiségét tárolja.
- **int work**: Az adott játékos munkájának (szebben megfogalmazva: munkára fordítható energiájának) egységeit tárolja.
- **Tool tools**: Tárolja a játékosnál található tárgyakat.
- **Field actualField**: Tárolja, hogy az adott játékos melyik mezőn áll.

Metódusok

- **Result round()**: Elsőként beállítja a jelenlegi játékos work attribútumának értéket négy egységre, majd végigvárja (egy ciklusban) a játékos lépéseit (míg a work értéke nulla nem lesz, vagy véget nem ér a játék győzelem vagy halál miatt). Futás során az általunk választott cselekvésekhez szükséges függvényeket fogja meghívni. Ha az általa meghívott függvények OK-kal térnek vissza, akkor csökkenti a work értékét eggyel, majd ellenőrzi, hogy ezt követően nem csökkent-e nullára. Amennyiben nem, akkor folytatódik a ciklus futása, ellenben ha ez az érték nullára csökkent, akkor kilép a ciklusból, és a round() metódus OK értékkel tér vissza. Ha futás során bármely függvény DIE vagy WIN visszatérési értékkel rendelkezik, akkor a round() szintén kilép a ciklusából és ugyanazzal fog visszatérni amit kapott. (NOTHING hatására nem csökkenti a work attribútumot, és biztosan benne marad a ciklusban)
- **Result clean()**: Meghívja a *Field* osztály clean() metódusát. Amennyiben ez OK-kal tér vissza, akkor meghívja a tools attribútumban tárolt összes Tool példány clean(Field) függvényét. Végül visszatér azzal, amivel a *Field* osztály először meghívott clean() függvénye tért vissza.
- **Result assemble()**: A game attribútumban tárolt Game-re meghívja a getPlayerNumber() függvényt. A visszakapott értéket átadja az általa hívott a haveAllPlayers() metódusnak. Ha ez FALSE értékkel tér vissza, akkor ő OK-kal fog. TRUE esetén viszont meghívja a HaveAllParts() függvényt, amely ha TRUE-val tér vissza, akkor az assemble() WIN-nel fog. Amennyiben a haveAllParts() metódus FALSE értékkel tér vissza, akkor az assemble() OK-kal fog. (Ezekben az esetekben azért tér vissza OK-kal, mert a feleslegesen megpróbált összeszerelés is munkának számít).
- ***abstract void specialSkill()***: Absztrakt függvény. Az Eskimo vagy az Explorer osztály specialSkill() függvénye hívódik meg.

- **Result changeField(Field f)**: Meghívja az actualfieldben tárolt mezőre a *Field* osztály leaveField(Player) függvényét. Ezek után beállítja a jelenlegi játékos actualfield nevű attribútumának értékét a megkapott mezőre. Ezt követően meghívja a *Field* osztály stepOn(Player) függvényét, és azzal fog visszatérni, amivel az általa hívott metódus visszatért.
- **Result helpMe()**: Először a swim(Field, Player) metódus kerül meghívásra minden egyes eszközre. Ezt követően a visszatérési értékek kerülnek vizsgálatra. Ha ezek közül bármelyik nem NOTHING értéket vesz fel, akkor azzal az értékkel tér vissza a helpMe() is. Ha pedig mindegyik NOTHING-gal tér vissza, akkor minden irányt megvizsgál a checkNeighbour(Direction) metódussal. Amennyiben a visszatérési érték nem NULL, akkor meghívódik a canHelp() függvény a megkapott referenciára. Ebben az esetben amivel ez a metódus tér vissza, azzal fog a helpMe() is.
- **Result decreaseHeat()**: Csökkenti a játékos heat nevű attribútumának értékét eggyel, majd megvizsgálja, hogy mennyi a heat értéke. Amennyiben ez nullára csökkent, akkor DIE, minden más esetben pedig OK visszatérési értéke lesz.
- **void addTool(Tool t)**: A listához hozzáadja a megkapott eszközt.
- **Result move(Direction d)**: Legelőször a megkapott irányra meghívja a checkNeighbour(Direction) metódust. Ezt követően az előbb hívott függvény visszatérési értékét átadva kerül a changeField(Field) meghívásra. Amivel ez visszatér, azzal fog a move(Direction) is.
- **Result increaseHeat()**: Legelőször megvizsgálja, hogy az adott játékos heat attribútumának értéke a maximális érték alatt van-e. Amennyiben igen, akkor megnöveli eggyel az értékét, majd OK visszatérési értéket ad. Ellenkező esetben kimarad a növelés, és NOTHING értékkel tér vissza.
- **void addPart(FlareGun f)**: Meghívja a Game osztály addPart(FlareGun) függvényét.
- **List<Tool> getTools()**: Visszaadja az eszközöket tartalmazó listát, tehát a tools attribútumát.

4.3.14 Result

Felelősség

Enumeration osztály, amely a meghívott függvények futása alatt bekövetkező eseményeket reprezentálja. A legtöbb függvény visszatérési értéke ezen enumeráció értékei közül vesz fel egyet.

Literálok

- **WIN**: A függvény futása során megnyerték a játékosok a játékot (összeszerelték és elsütötték a jelzőpisztolyt).
- **DIE**: A metódus futása során meghalt (legalább egy) játékos, így a játékosok elvesztették a játékot.
- **NOTHING**: A függvény eredeti célja szerinti munka nem végződött el (pl.: volt hó a mezőn nem lehetett tárgyat felvenni).
- **OK**: Az előző elemek közül egyik sem.

4.3.15 Rope

Felelősség

A kötél felvételére, illetve a köteles kimentés kezelésére szolgáló osztály.

Ősosztályok

- Tool → Rope

Interfészek

- Nincsenek.

Attribútumok

- Nincsenek.

Metódusok

- **boolean isSame(Rope r)**: Mindig TRUE értékkel tér vissza (hiszen csak akkor hívódik meg, ha egy Rope példány szeretné magát összehasonlítani vele).
- **Result help(Field f, Player p)**: Meghívja a paraméterként megkapott Player példány changeField(Field) metódusát, átadva neki a paraméterként a kapott Field példányt. A changeField függvény visszatérési értéke lesz a help függvény visszatérése is.

4.3.16 Shovel

Felelősség

Az ásó felvételének, illetve az ásóval rendelkező játékosok hóréteg ellapátolásának kezelésére szolgáló osztály.

Ősosztályok

- Tool → Shovel

Interfészek

- Nincsenek.

Attribútumok

- Nincsenek.

Metódusok

- **boolean isSame(Shovel s)**: Mindig TRUE értékkel tér vissza (hiszen csak akkor hívódik meg, ha egy Shovel példány szeretné magát összehasonlítani vele).
- **void clean(Field f)**: Akkor hívódik meg, ha az ásást végző játékosnál van ásó. Ekkor ez a függvény meghívja a *Field* clean() metódusát, ezzel még egy réteget ellapátolva arról (persze, ha ez lehetséges). Void visszatérésű, mivel nincs jelentősége, hogy ez a művelet sikerült-e vagy sem.

4.3.17 Tool

Felelősség

Az eszközök felvételének, illetve a velük kapcsolatos interakciók (ásás, köteles kimentés, búvárruha használatával történő kimenekülés) kezelésére szolgáló osztály.

Ősosztályok

- Nincsenek.

Interfészek

- Item

Attribútumok

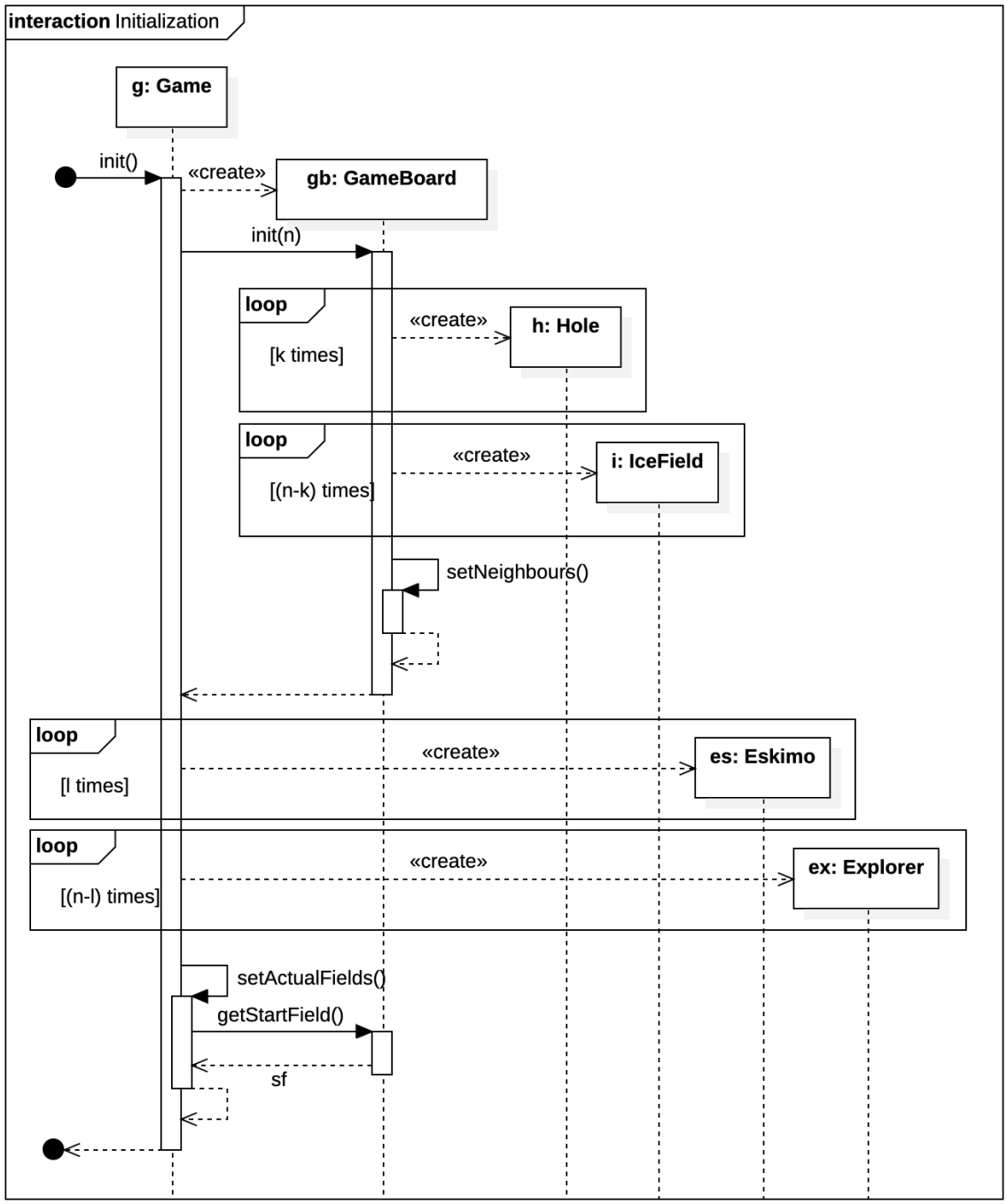
- Nincsenek.

Metódusok

- **Result pickMeUp(Player p)**: Legelőször a Player osztály getTools() függvénye kerül meghívásra, mely a játékosnál lévő eszközöket tartalmazó listával tér vissza. Ezt követően meghívja a lista minden elemére a Tool osztály isSame(Item) metódusát. Ezután ezeknek a visszatérési értékei kerülnek vizsgálat alá. Amennyiben minden függvény hívást követően csak FALSE visszatérési értékeket kapunk, akkor meghívásra kerül a Player osztály addItem(Item) metódusa, majd ezt követően OK-kal tér vissza. Különben pedig NOTHING lesz a visszatérési érték.
- **boolean isSame(Tool t)**: Megvizsgálja a megkapott eszközre, hogy az adott játékos rendelkezik-e már vele.
- **virtual void clean(Field f)**: Virtuális, üres függvény.
- **virtual Result swim(Field f, Player p)**: Virtuális függvény, ami NOTHING értékkel tér vissza.
- **virtual Result help(Field f, Player p)**: Virtuális függvény, ami NOTHING értékkel tér vissza.

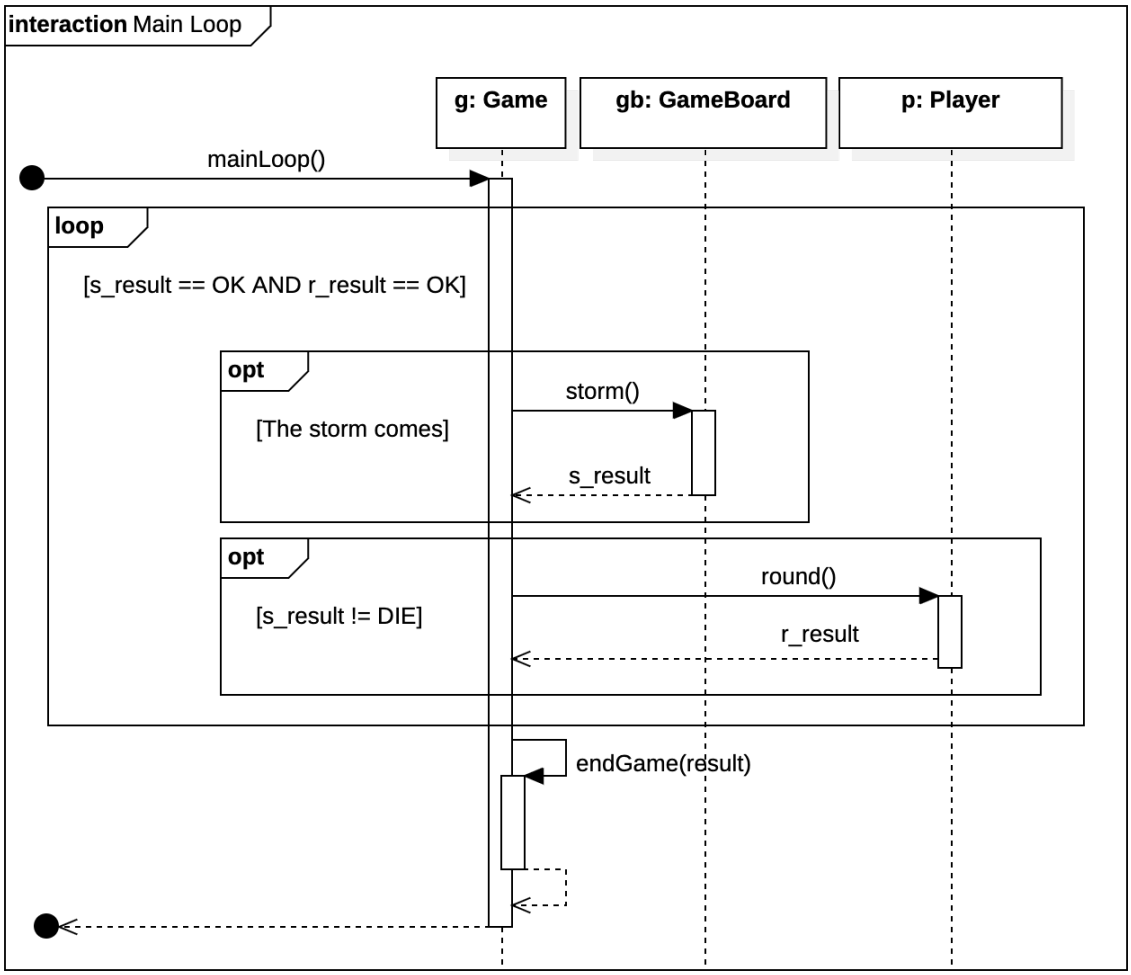
4.4 Szekvencia diagramok

4.4.1 Initialization



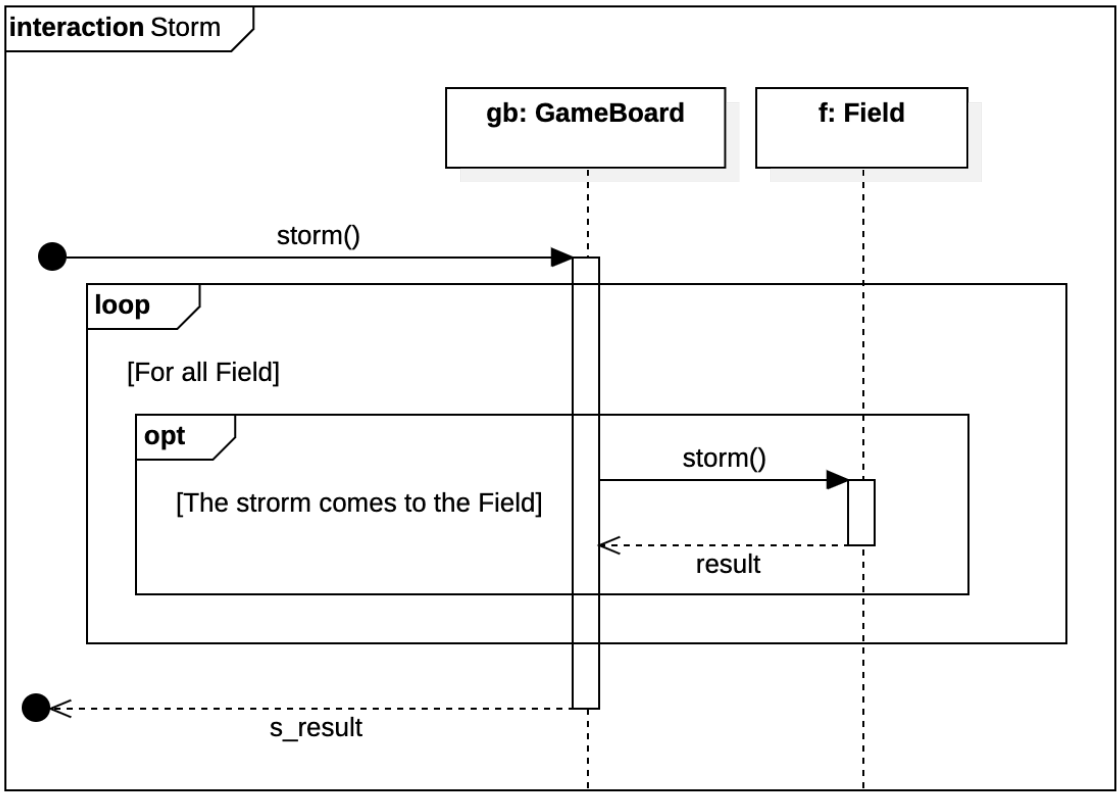
A gb példány megegyezik a Game osztály board attribútumában tárolt példánnyal, tehát ismerik egymást.
Az n a játékosok számát adja meg.

4.4.2 Main Loop



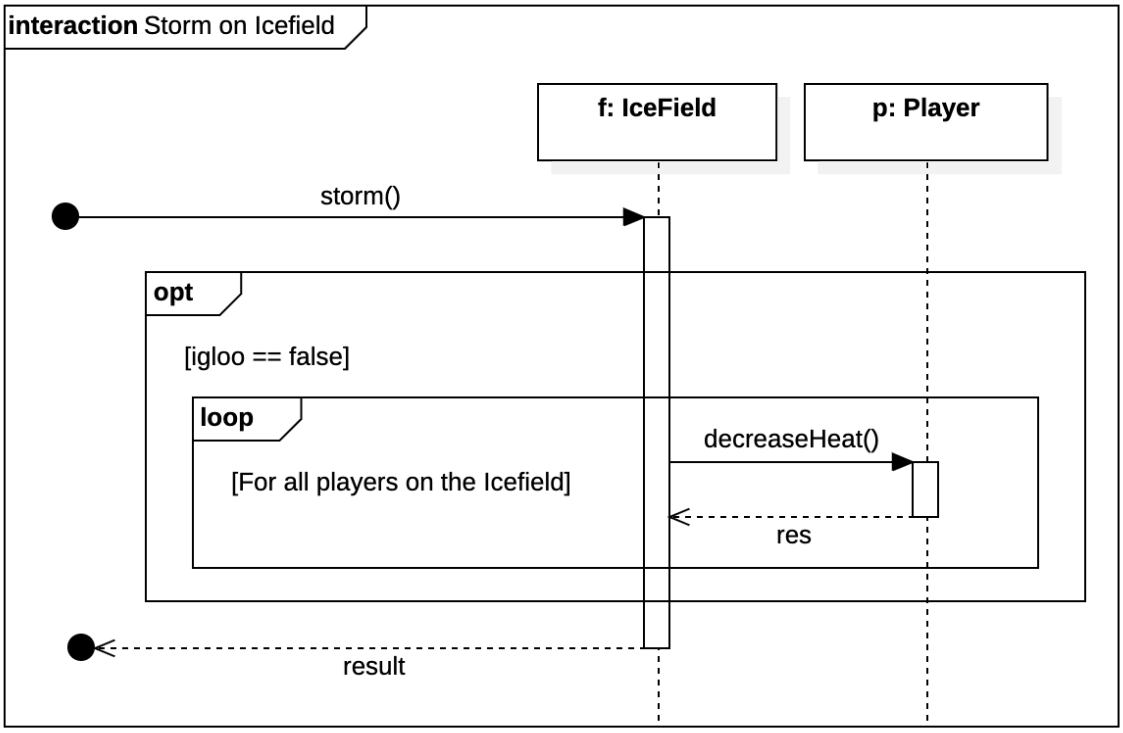
A gb példány megegyezik a Game osztály board attribútumában tárolt példánnyal, tehát ismerik egymást.
A Player példányokat pedig a players attribútumban tárolja, tehát azokat is ismeri.

4.4.3 Storm



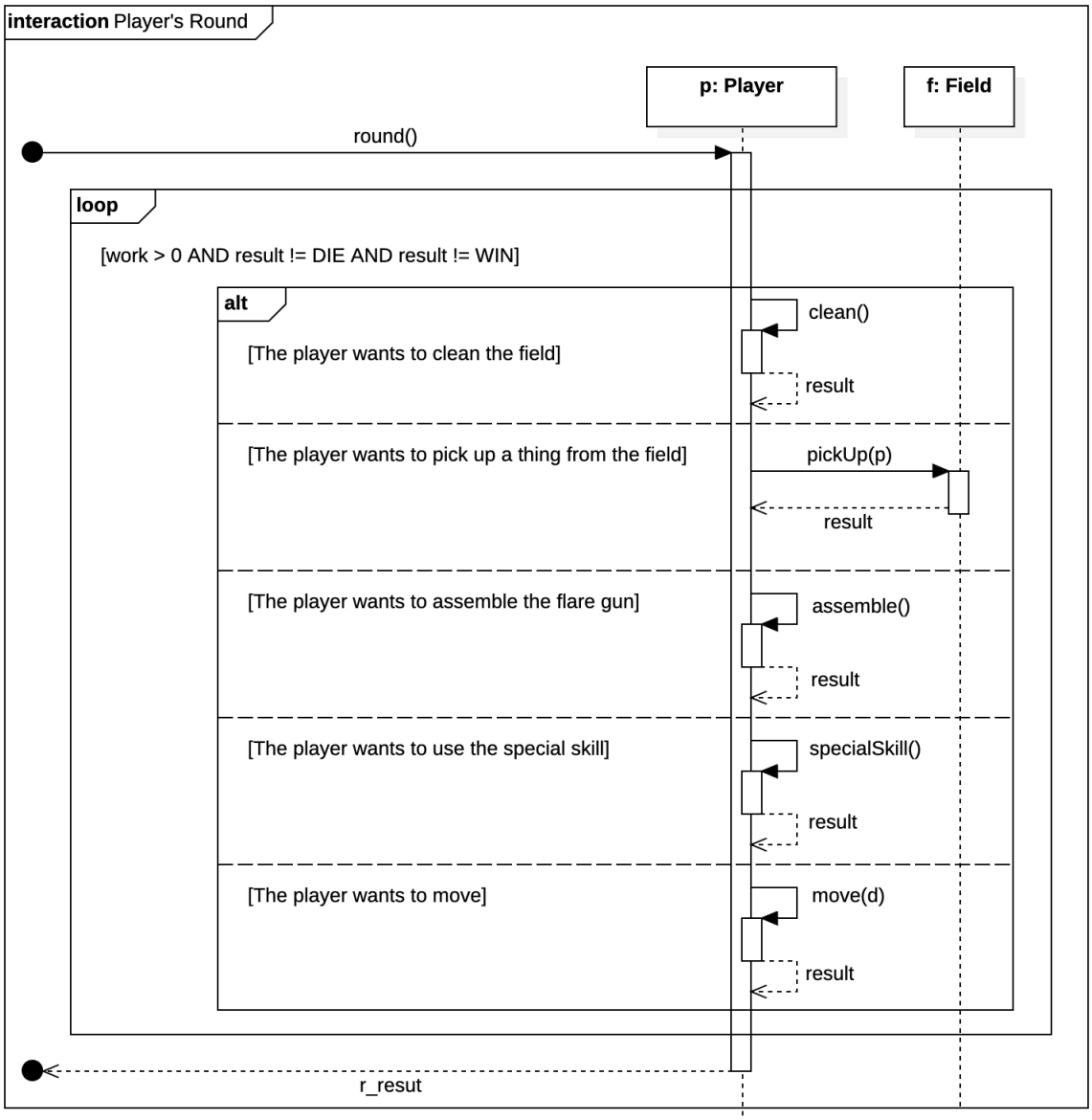
Az Field példányokat a GameBoard osztály a fields attribútumában tárolja, tehát ismerik azokat.

4.4.4 Storm on Icefield



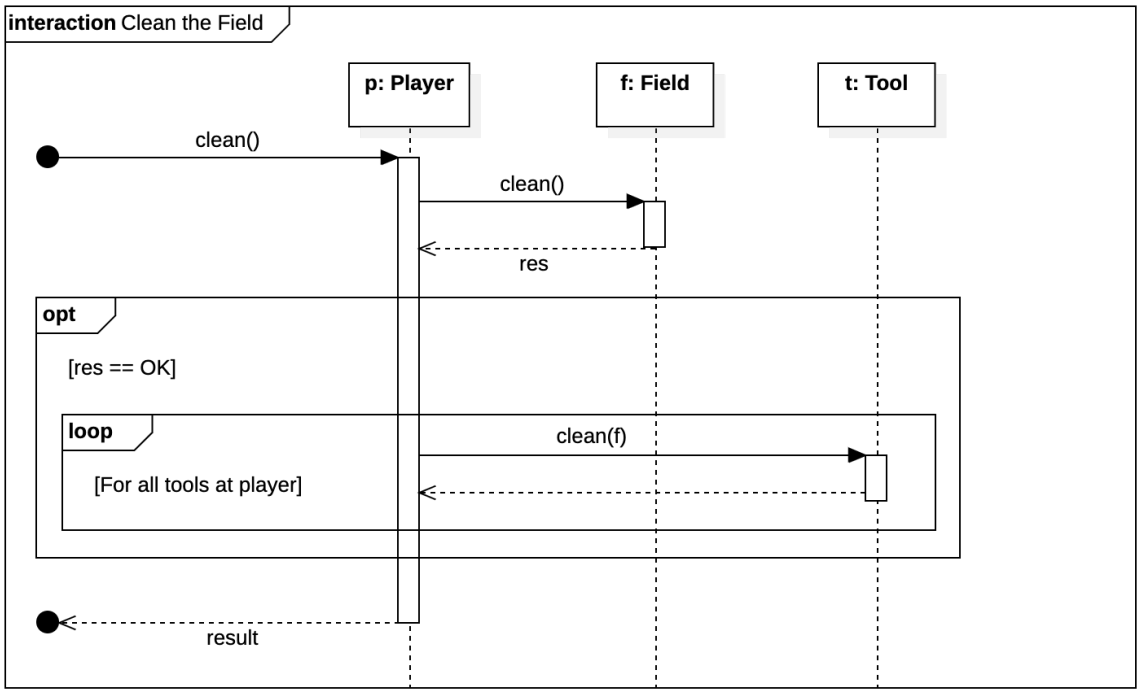
A Player példányokat az IceField osztály a players attribútumában tárolja, tehát ismerik azokat.

4.4.5 Player's Round



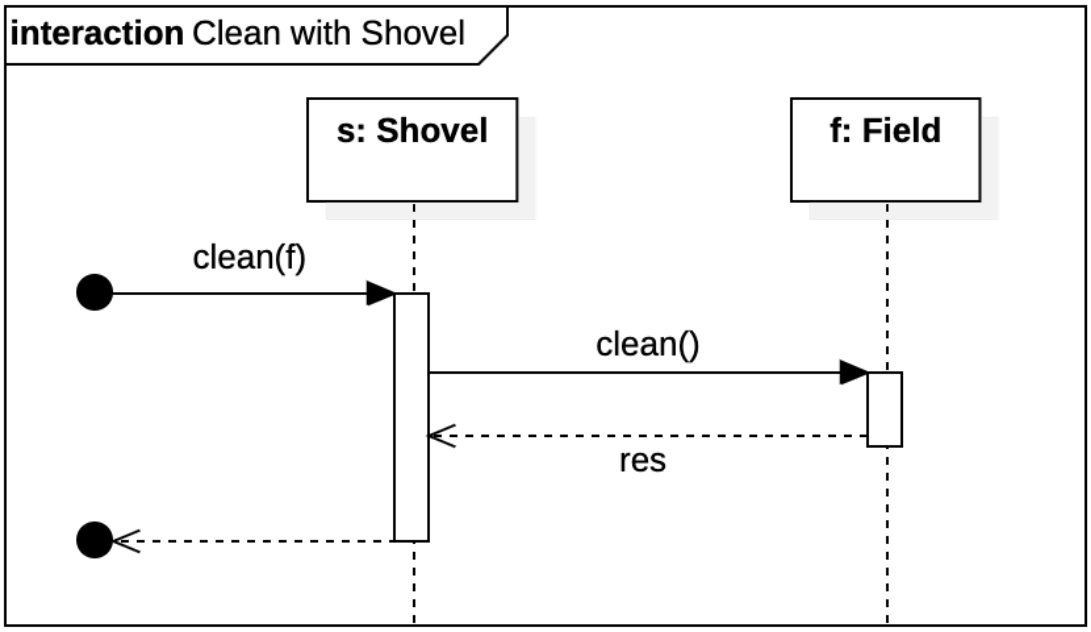
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri.

4.4.6 Clean the Field



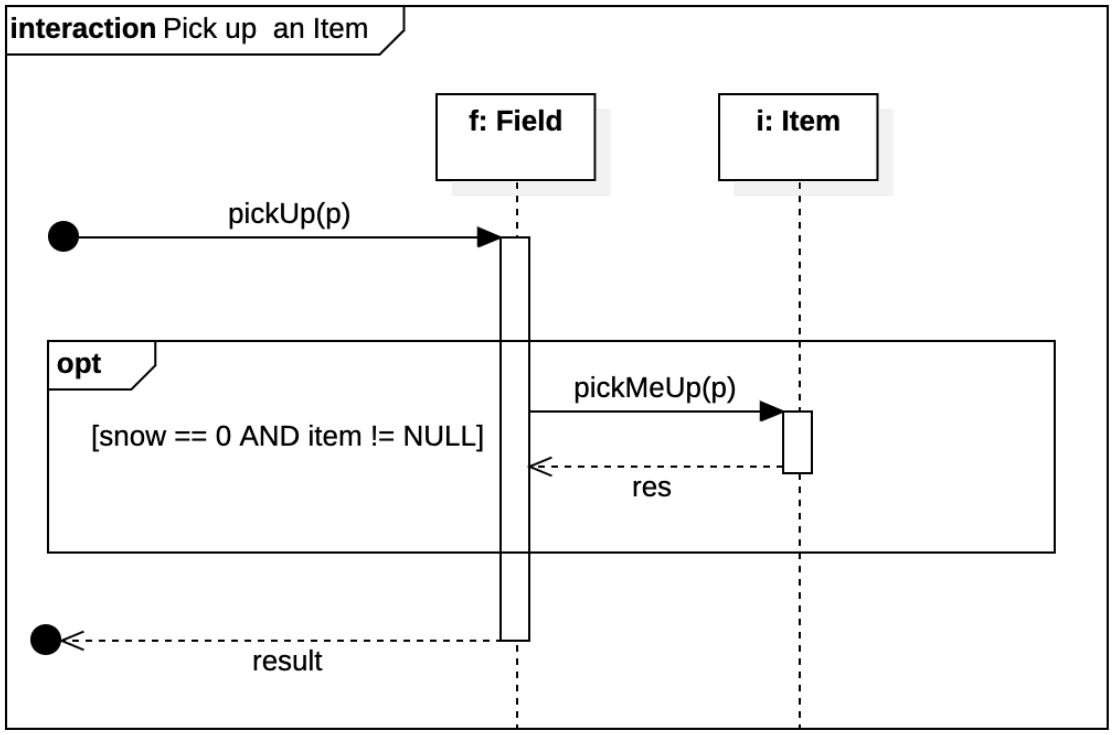
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri. A t Tool-okat a játékos a tools attributumában tárolja.

4.4.7 Clean with Shovel



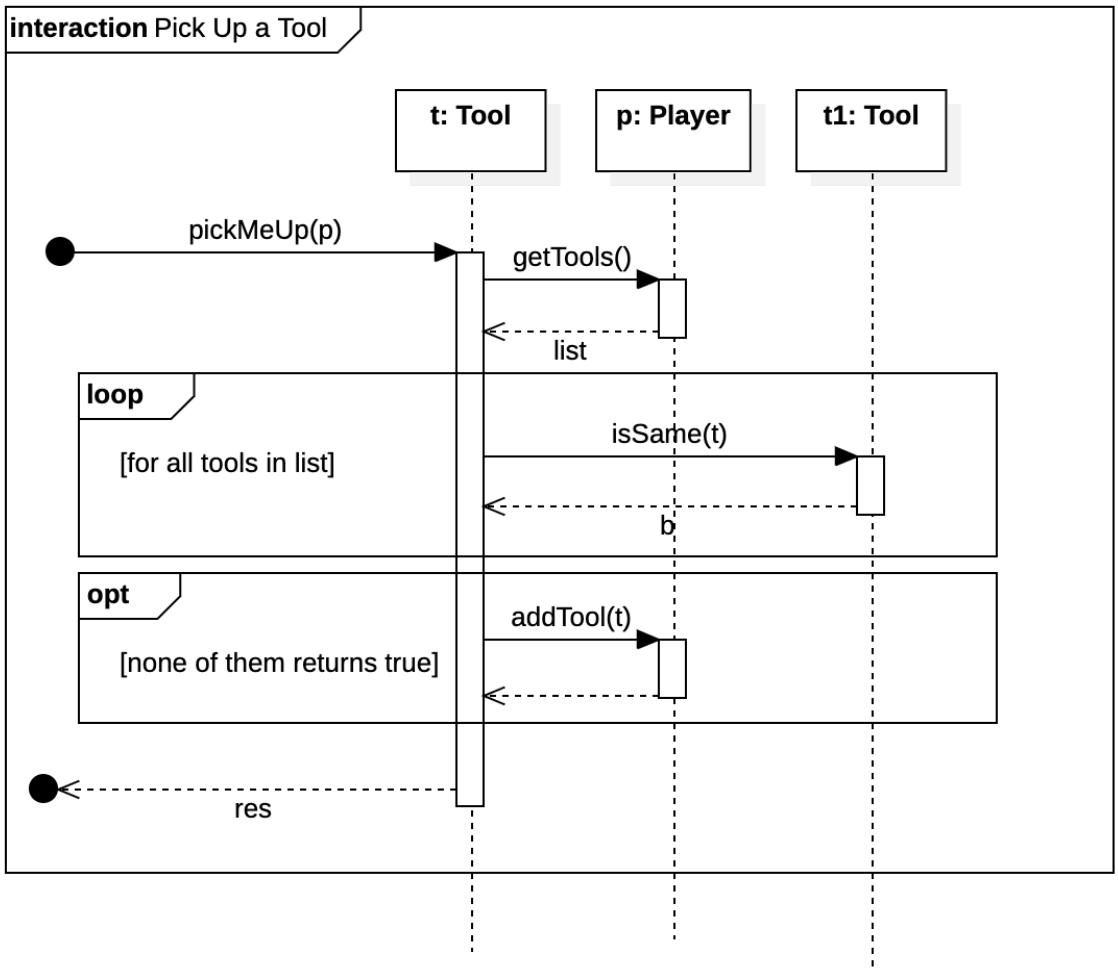
Az f példányt a clean(Field) függvény attribútumában megkapja a Shovel, tehát ismeri.

4.4.8 Pick up an Item



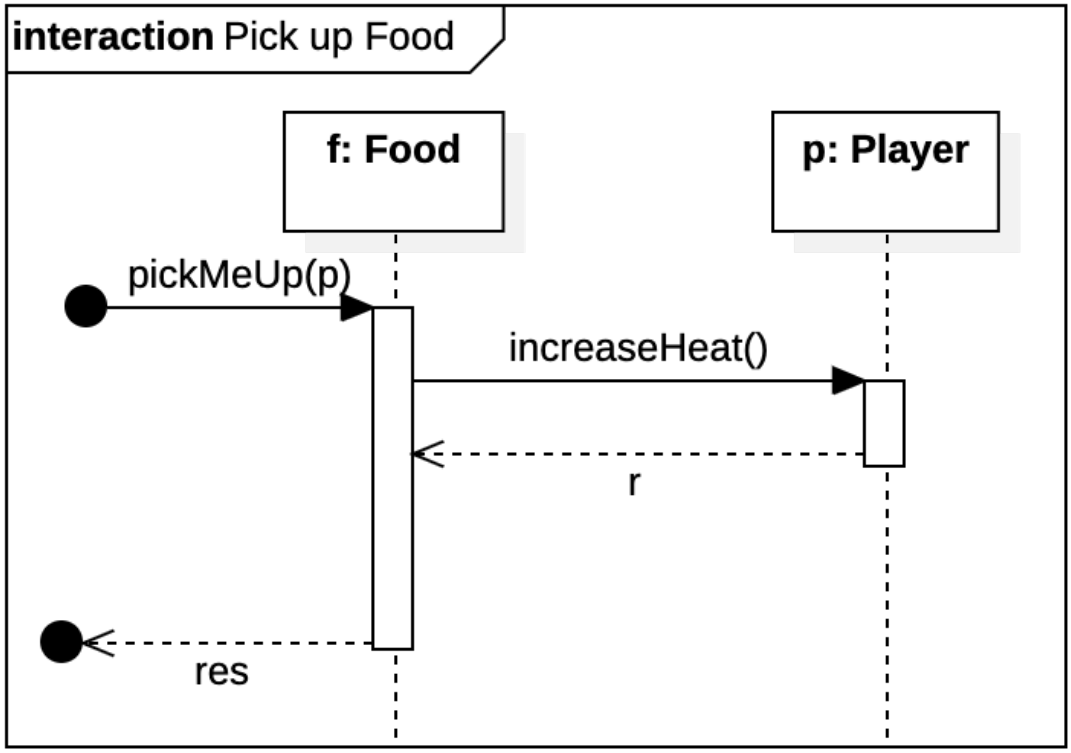
A i Item-et a mező az item attributumában tárolja, tehát ismeri.

4.4.9 Pick up a Tool



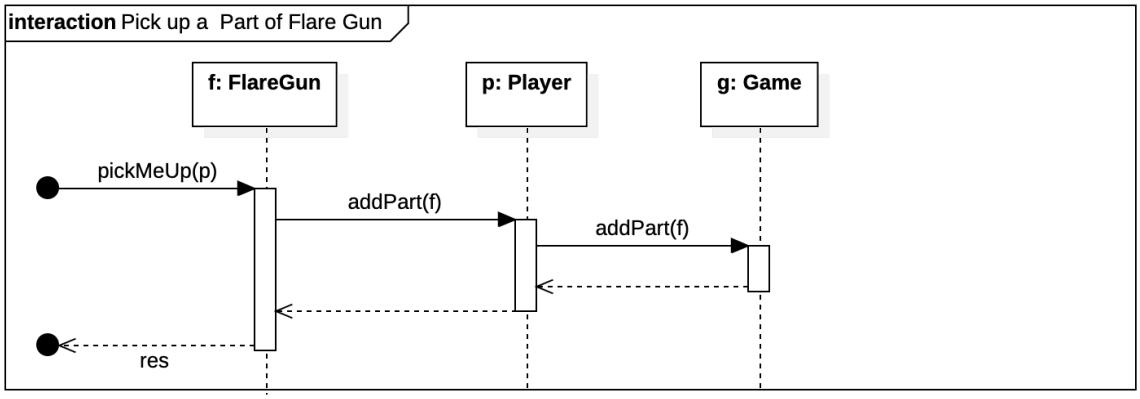
A p példányt a pickMeUp(Player) függvény attribútumában megkapja a Tool, tehát ismeri. Az eszközöket tartalmazó listát pedig a getTools() függvénnyel kéri le a Playertől, így a t1 Tool-okat is ismeri.

4.4.10 Pick up Food



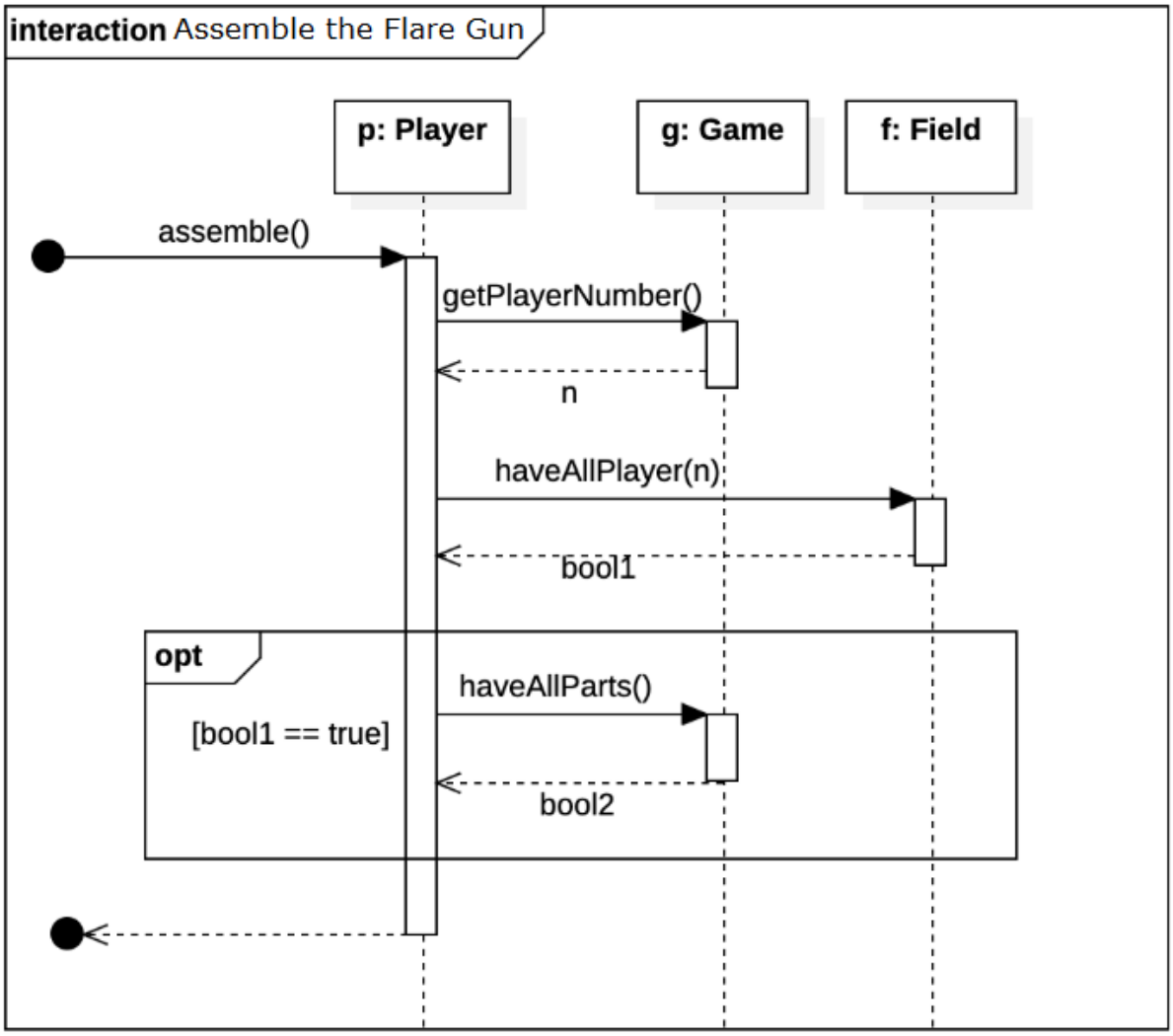
A p példányt a pickMeUp(Player) függvény attribútumában megkapja a Food, tehát ismeri.

4.4.11 Pick up a Part of Flare Gun



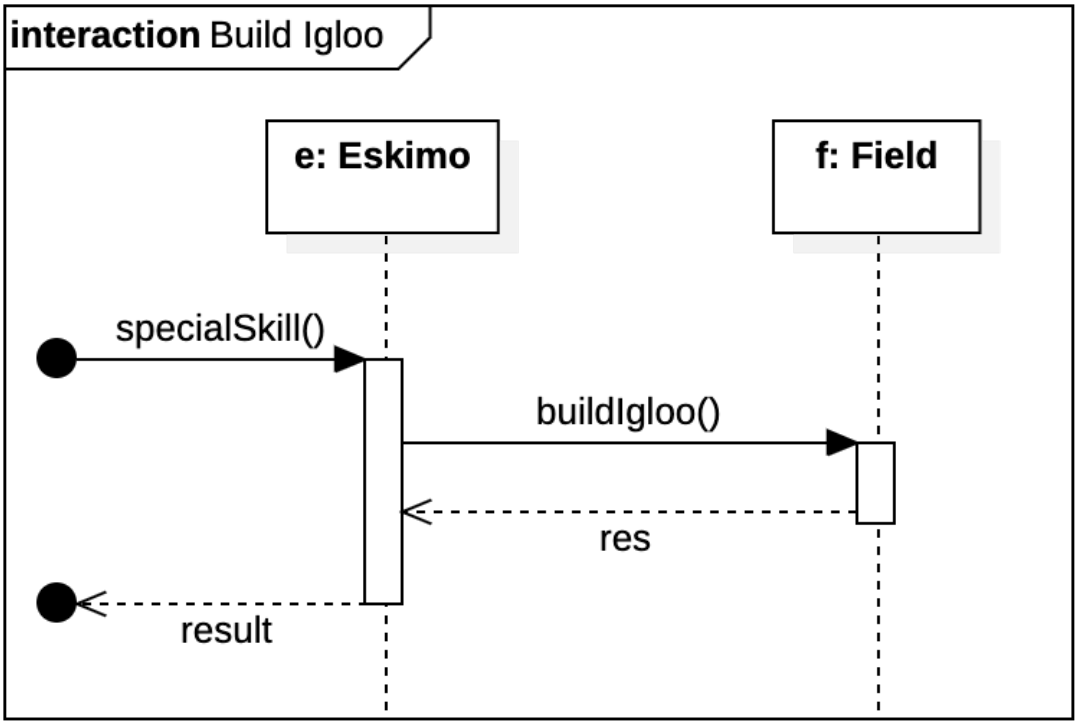
A p példányt a pickMeUp(Player) függvény attribútumában megkapja a FlareGun, tehát ismeri. A Player pedig a game attribútumában tárolja a meghívott g Game példányt.

4.4.12 Assemble the Flare Gun



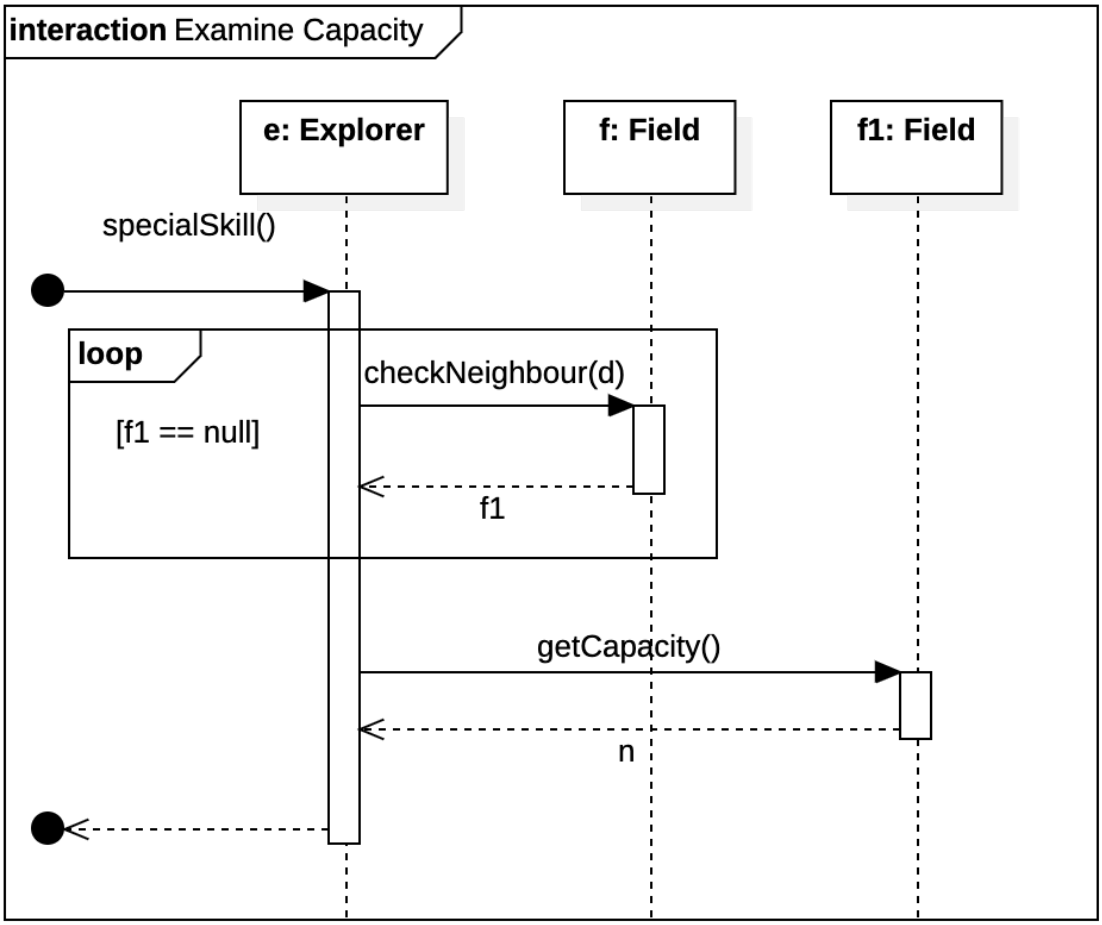
A g példány megegyezik a Player osztály game attribútumában tárolt példánnyal, tehát a Player ismeri.
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ezt is ismeri a Player.

4.4.13 Build Igloo



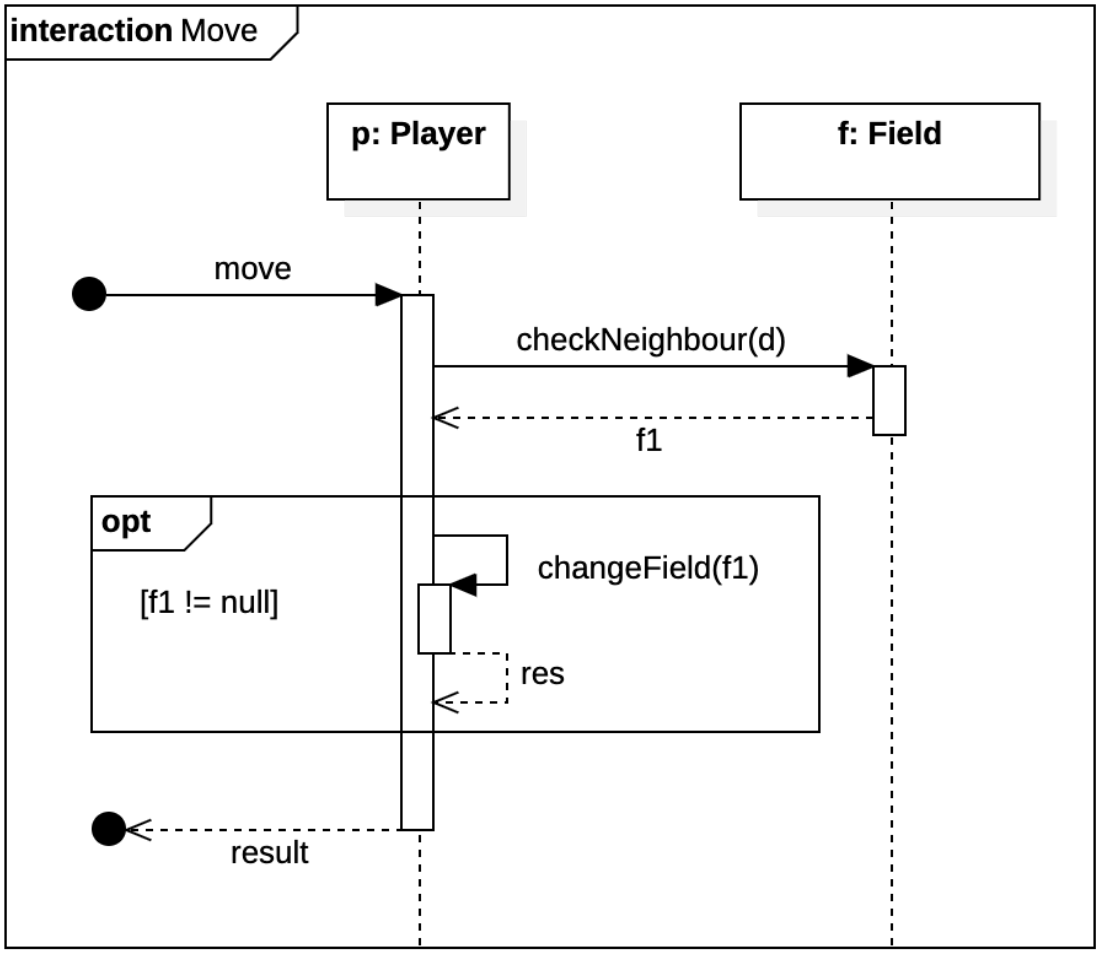
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri a Player.

4.4.14 Examine Capacity



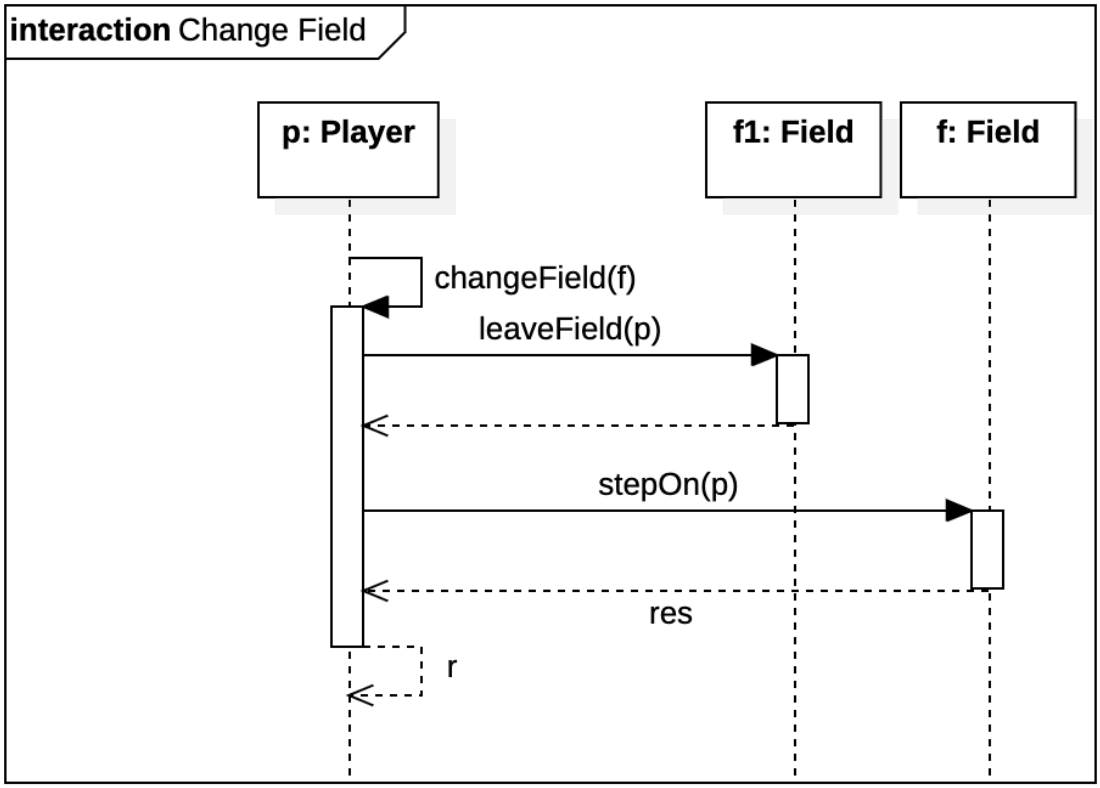
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát a Player ismeri.
A checkNeighbour(Direction) függvény pedig visszatér az f1 referenciájával, tehát azt is ismeri a Player osztály.

4.4.15 Move



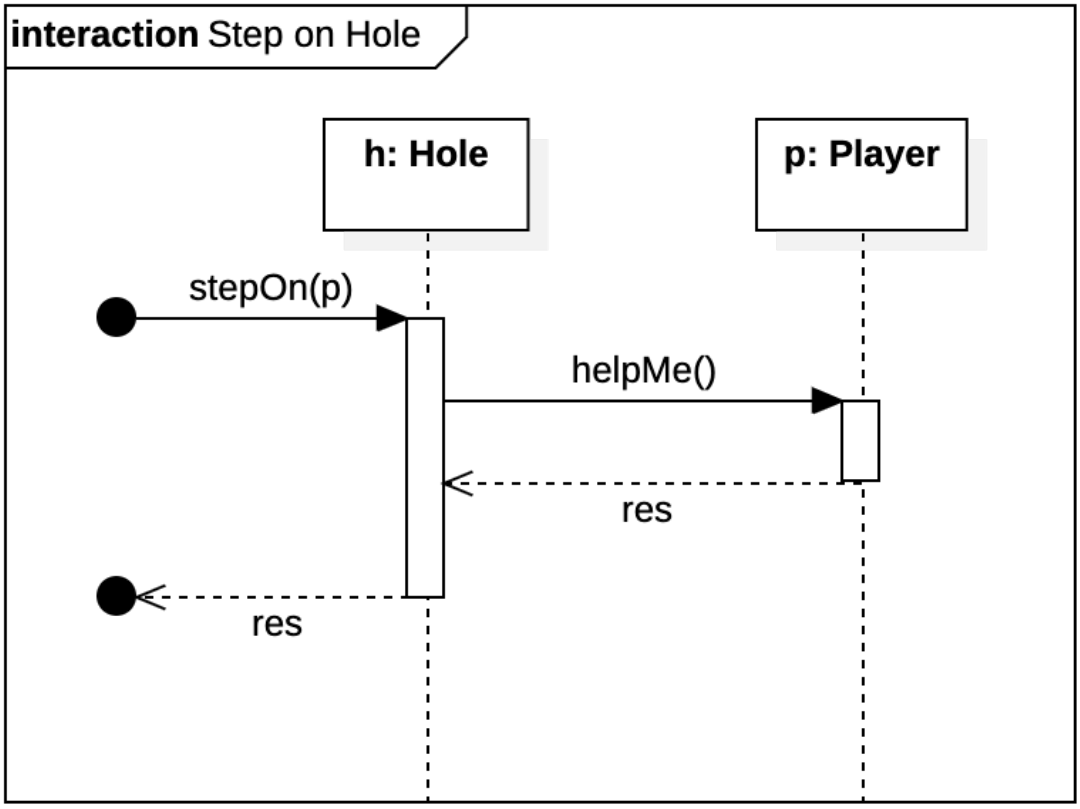
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát a Player ismeri.

4.4.16 Change Field



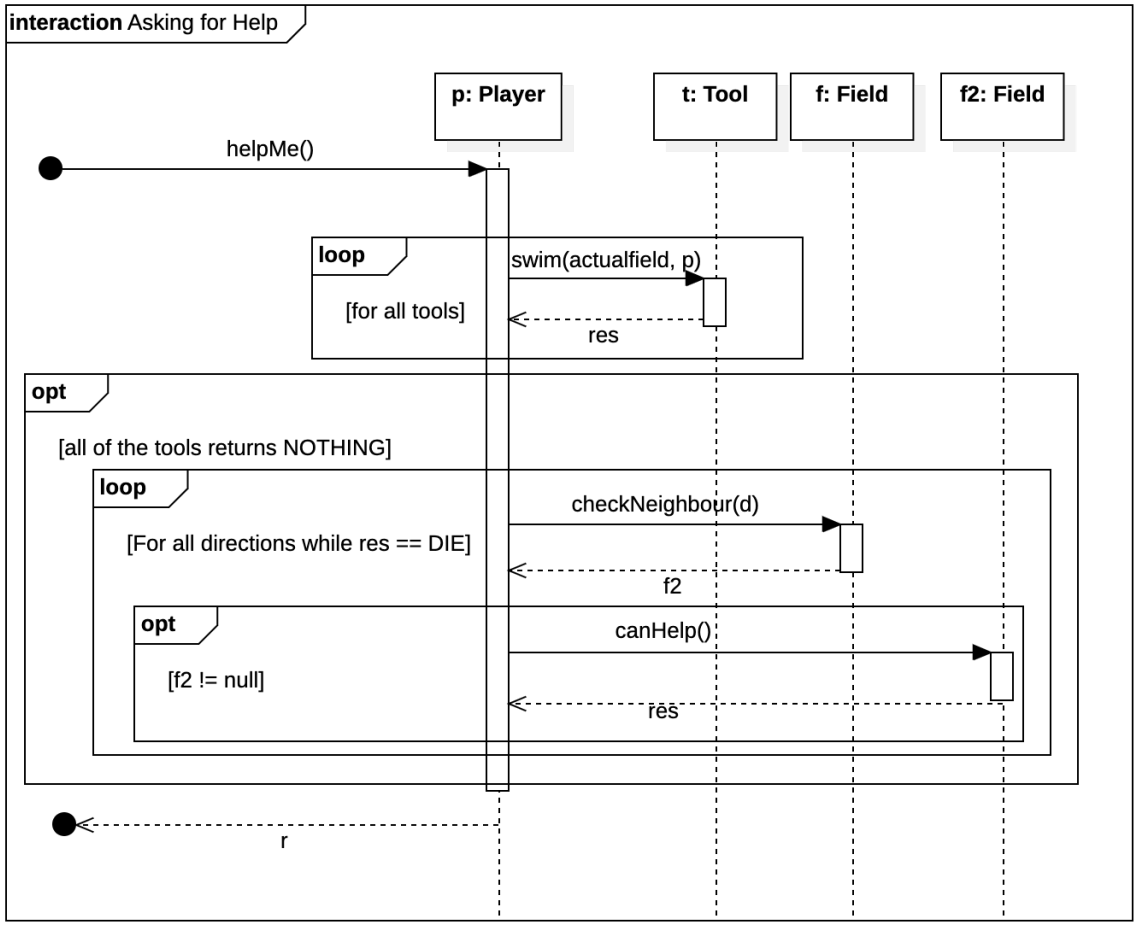
Az f1 példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást. Az f példányt pedig a changeField(Field) függvény attribútumában kapja meg a Player, így azt is ismeri.

4.4.17 Step on Hole



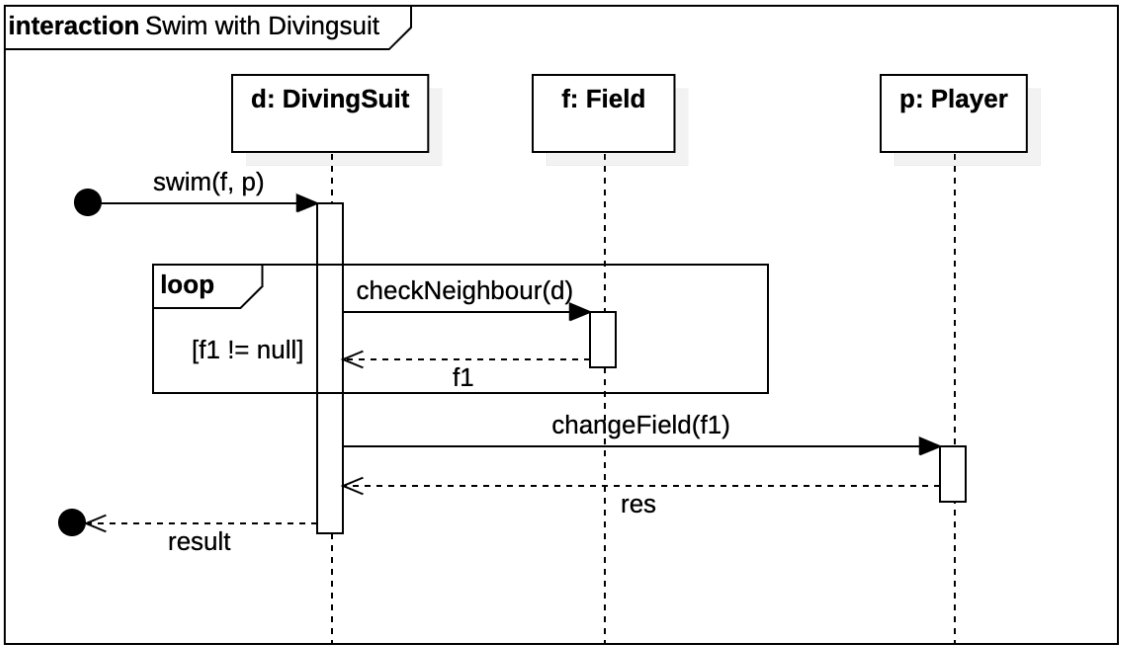
A p példányt a stepOn(Player) függvény attribútumában megkapja a Hole, tehát ismeri.

4.4.18 Asking for Help



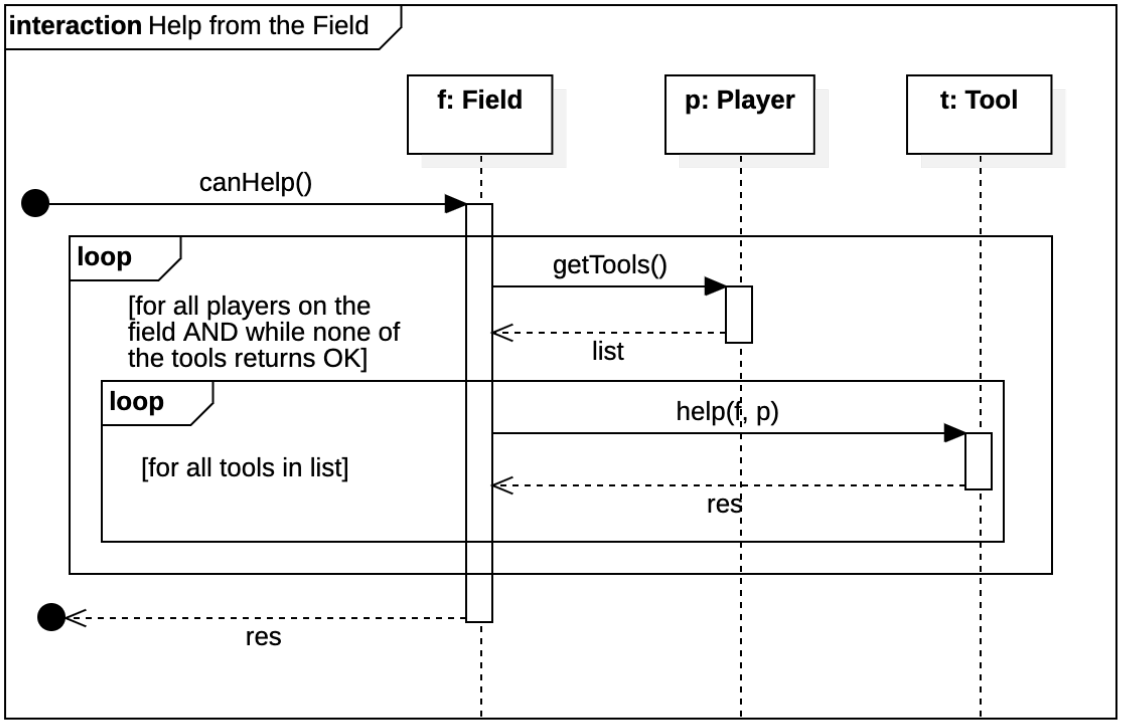
A Player a tools attribútumában tárolja az eszközöket, tehát ismeri a t példányokat.
Az f példány pedig megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát a Player azt is ismeri.
Az f2 példányok pedig megegyeznek a checkNeighbour(Direction) függvény visszatérési értékeivel, tehát azokat is ismeri a Player.

4.4.19 Swim with Divingsuit



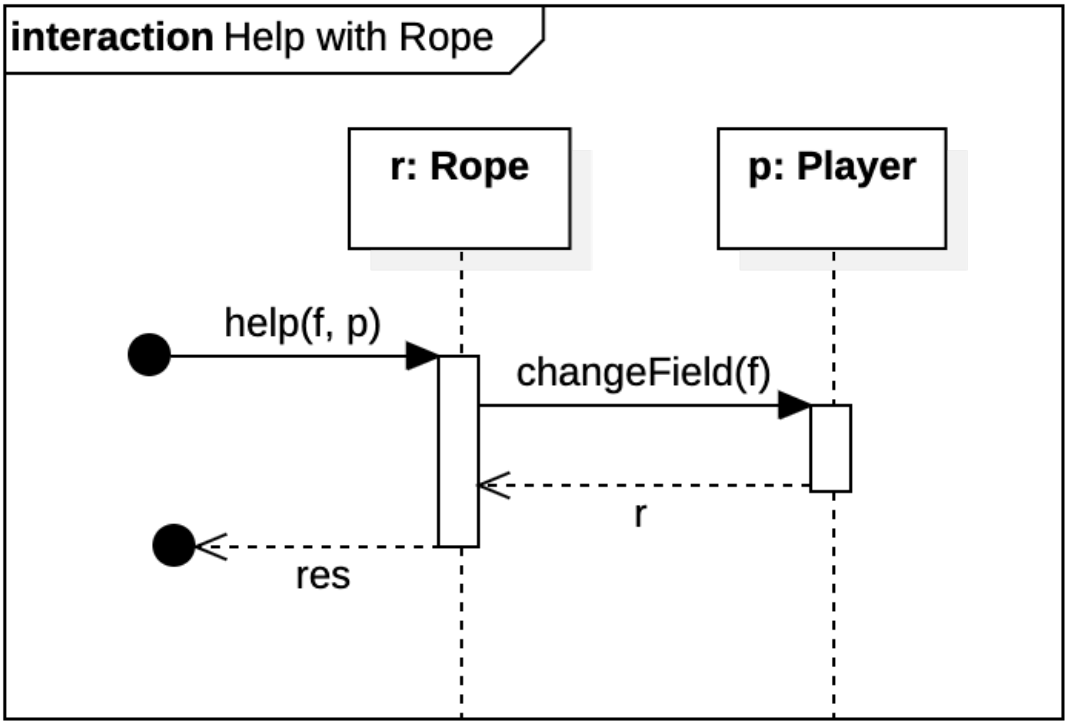
Az f és p példányokat a swim(Field, Player) függvényben megkapja a DivingSuit, tehát ismeri azokat.

4.4.20 Help from the Field



A Player példányokat a *Field* osztály a players attribútumában tárolja, tehát ismerik azokat. Az eszközöket tartalmazó listát pedig a getTools() függvénnyel kéri le a Playertől, így a t Tool-okat is ismeri.

4.4.21 Help with Rope



A p példányt a help(Field, Player) függvény attribútumában megkapja a Rope, tehát ismeri.

4.5 State-chartok

Egyik osztály sem tartalmaz olyan állapotokat, melyek szemléltetése célszerű lenne state-charttal.

4.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2020. 03. 06. 15:00	4 óra	Fábián	Osztálydiagram (4.2) és Szekvencia diagramok újragondolása, és újrarajzolása (4.4).
2020. 03. 07. 17:00	1,5 óra	Fábián Ilosvay	Konzultáció: Az átalakított diagramok átnézése, információk átadása a dokumentáláshoz.
2020. 03. 08. 17:00	5 óra	Ilosvay	Osztályleírások (4.3) kiegészítése, javítása.
2020. 03. 08. 21:30	2 óra	Fábián	Osztályok leírásának (4.3) ellenőrzése, javítása
2020. 03. 08. 23:30	1 óra	Szabó K	Szekvenciadiagramok elrendezése (4.4) a dokumentumban.
2020. 03. 09. 8:00	2 óra	Ilosvay	Teljes dokumentum ellenőrzése, konzisztencia vizsgálata, javítása. Majd végső formázása és nyomtatás.